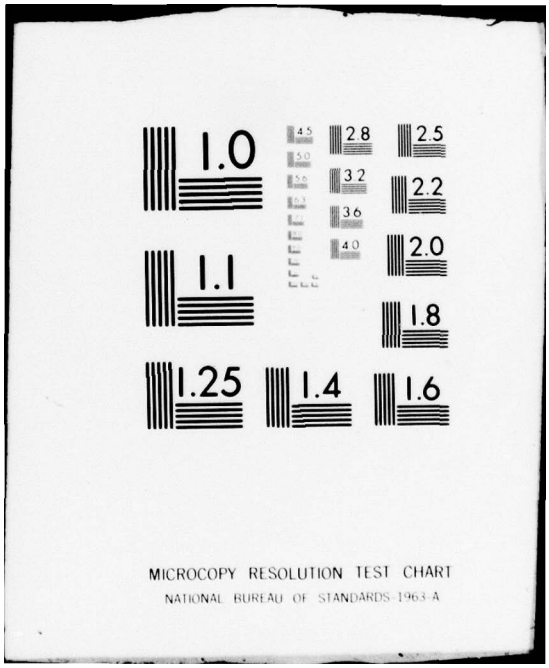
AD-A073 795 STANFORD UNIV CALIF DEPT OF OPERATIONS RESEARCH F/G 12/1
A SIMPLE ALTERNATIVE TO THE OUT-OF-KILTER ALGORITHM.(U)
MAY 79 N ZADEH N00014-75-C-0493
UNCLASSIFIED TR-35 NL

| OF |
AD
A073 795

END
DATE
FILMED
10-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(12)

LEVEL II

A SIMPLE ALTERNATIVE TO THE OUT-OF-KILTER ALGORITHM

BY

NORMAN ZADEH

TECHNICAL REPORT NO. 35

MAY 31, 1979

PREPARED UNDER
OFFICE OF NAVAL RESEARCH CONTRACT
N00014-75-C-0493 (NR-042-264)

DEPARTMENT OF OPERATIONS RESEARCH
STANFORD UNIVERSITY
STANFORD, CALIFORNIA

A SIMPLE ALTERNATIVE TO THE OUT-OF-KILTER ALGORITHM

by

Norman Zadeh

Technical Report No. 35

31 MAY 1979

44 p.

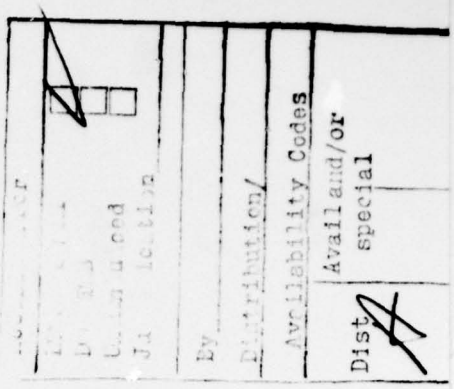
TR-35

Prepared Under
Office of Naval Research Contract N00014-75-C-0493*

NSF-ENG76-12266

Department of Operations Research
Stanford University
Stanford, California

*Also partially supported by National Science Foundation Grant ENG 76-12266.



A Simple Alternative to the Out-of-Kilter Method

Abstract

It is shown that any problem solvable by the Out-of-Kilter method may be simplified to an ordinary minimum cost flow problem (meaning a problem with zero lower bounds). To perform the simplification, one considers the equivalent problem of optimally augmenting the original flows and then eliminates the lower bounds from this problem. In linear programming terms, as many as $|A|$ constraints are eliminated without adding new variables, where $|A|$ is the number of arcs.

When the simplified problem has non-negative arc costs, we show that Out-of-Kilter will solve it by augmenting along a sequence of shortest paths, and hence is just another implementation of such methods. In particular, Out-of-Kilter combines the Ford-Fulkerson labeling procedure with a $O(n^3)$ version of Dijkstra that computes shortest paths by setting the costs of negative arcs to zero.

Examples are presented which show that Out-of-Kilter may behave pathologically when some arc costs are negative. For example, Out-of-Kilter may take $O(n^5)$ steps to find a negative cycle or compute a shortest path.

The following procedure is suggested to replace the Out-of-Kilter method: Transform to a minimum cost flow problem, eliminate negative cycles if any, then efficiently augment along a sequence of shortest paths.

Introduction

The Out-of-Kilter method, first introduced in 1960-61 independently by D. R. Fulkerson and George Minty, is a general technique which can solve any minimum cost flow[1] or circulation problem and perform sensitivity studies starting with any node numbers and with any solution, feasible or not, as long as the solution satisfies conservation of flow. When applied to minimum cost flow problems, the Out-of-Kilter method is called the Primal Dual method; when applied to assignment problems, it is called the Hungarian method.

We show here that any Out-of-Kilter problem may be solved by a simple transformation to a minimum cost flow problem with the desirable property that the required level of s-t flow may be small, so the problem can be solved with relatively few augmentations along cheapest paths (after any negative cycles are eliminated). The transformation utilizes a trick from Ford and Fulkerson [4] and is essentially a change of variables. It eliminates as many as $|A|$ constraints from the standard L.P. formulation.

When the simplified problem has non-negative arc costs, Out-of-Kilter will solve it by augmenting along a sequence of shortest paths, and hence is just another implementation of such methods. We will call any such method a "Path" algorithm.

We argue that Out-of-Kilter consists of essentially two routines: the old Ford-Fulkerson maximum flow algorithm, and an inefficient $O(n^3)$ version of Dijkstra that computes shortest paths by setting the costs of all "can increase" and "must increase" arcs to zero. Roughly speaking, this means that Out-of-Kilter sets the costs of negative arcs to zero when computing shortest paths.

---

[1] By "minimum cost flow problem", we mean one with zero lower flow bounds. Circulations may have positive lower bounds.

When the simplified problem has some negative arcs, as might occur during a sensitivity study when arc costs are changed, we show that Out-of-Kilter may behave pathologically by augmenting along a sequence of incorrect paths when only one augmentation is necessary. Out-of-Kilter's procedure for initially computing shortest paths or finding negative cycles is shown to have a worst case behavior of $O(n^5)$, as compared to $O(n^3)$ for other methods.

Procedures for converting any minimum cost flow problem to one with non-negative arc costs are outlined [2], [7].

A future paper will show that the Simplex method with an artificial start and most negative pivot rule is also a Path algorithm. In other words, previous comparisons of Simplex vs. Out-of-Kilter represented comparisons of Simplex with the best empirical pivot rule (not most negative) vs. Simplex (most negative) with a somewhat different flow routine (Ford-Fulkerson).

Proposed Method to Replace the Out-of-Kilter Algorithm

We will first give a brief description of the method and then go into more detail. The method may be broken down as follows:

1. Given an initial circulation $\tilde{f}$ which may violate upper and lower bounds, it suffices to find the circulation f such that $\tilde{f} + f$ is optimal. This is a minimum cost circulation problem in which initial flows are zero, and therefore violate only lower bounds.

2. Eliminate lower bounds from the second problem using a change of variables. This yields a minimum cost flow problem with a fictitious source s and sink t.

   Let v equal the sum of the capacities of the arcs adjacent to s (or to t).

3. Compute a minimum cost s-t flow in the minimum cost flow network of value v. If no flow of value v exists, the original problem has no feasible solution.

The optimal solution to the original problem is obtained by adding the minimum cost flow plus the lower bounds to the original flows.

Details

Let f_{ij} = flow on arc (i, j),

c_{ij} = capacity of arc (i, j),

ℓ_{ij} = lower bound on flow in arc (i, j),

d_{ij} = cost per unit of flow in arc (i, j),

N = node set, A = arc set,

$$\ell(N, i) = \sum_{j \in N} \ell_{ji}\ , \quad \ell(i, N) = \sum_{j \in N} \ell_{ij}\ ,$$

$$f(N, i) = \sum_{j \in N} f_{ji}\ , \quad f(i, N) = \sum_{j \in N} f_{ij}$$

1. Given a non-feasible flow $\tilde{f} = \{\tilde{f}_{ij}\}$ in a network $\tilde{G}$, it suffices to find the minimum cost circulation $f = \{f_{ij}\}$ such that $\tilde{f} + f$ is feasible. To do this, form a new network G with all zero flows as follows:

a) $\tilde{f}_{ij} > \tilde{c}_{ij}$. Insert arc (j, i) in G with $\ell_{ji} = \tilde{f}_{ij} - \tilde{c}_{ij}$,

$c_{ji} = \tilde{f}_{ij} - \tilde{\ell}_{ij}$. $f_{ji} = 0$.

Example: $\xrightarrow[\tilde{\ell}_{ij}\ \tilde{c}_{ij}\ \tilde{d}_{ij}\ \tilde{f}_{ij}]{3,\ 8,\ \$2,\ \tilde{9}}$ goes to $\xleftarrow[\ell_{ji}\ c_{ji}\ a_{ji}]{1,\ 6,\ -\$2}$

b) $\tilde{f}_{ij} < \tilde{\ell}_{ij}$. Insert arc (i, j) in G with $\ell_{ij} = \tilde{\ell}_{ij} - \tilde{f}_{ij}$,

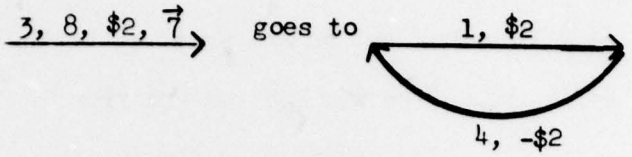
$c_{ij} = \tilde{c}_{ij} - \tilde{f}_{ij}$, $f_{ij} = 0$.

Example: $\xrightarrow{3,\ 8,\ \$2,\ \tilde{1}}$ goes to $\xrightarrow{2,\ 7,\ \$2}$

c) $\tilde{\ell}_{ij} \le \tilde{f}_{ij} \le \tilde{c}_{ij}$. Insert arcs (i, j) and (j, i) with

$c_{ij} = \tilde{c}_{ij} - \tilde{f}_{ij}$, $c_{ji} = \tilde{f}_{ij} - \tilde{\ell}_{ij}$, $d_{ij} = \tilde{d}_{ij}$, $d_{ji} = -\tilde{d}_{ij}$,

$\ell_{ij} = \ell_{ji} = f_{ij} = f_{ji} = 0$.

Example: 

2. Eliminate lower bounds on all arcs as follows:

The minimum cost circulation problem for the current network is

1)
$$\text{minimize} \quad \sum_{(i,\, j) \in A} d_{ij} f_{ij}$$

$$\text{subject to} \quad f(i, N) - f(N, i) = 0 \qquad \forall\, i \in N$$

$$0 \le \ell_{ij} \le f_{ij} \le c_{ij} \qquad \forall\, (i, j) \in A\ .$$

Set $f'_{ij} = f_{ij} - \ell_{ij}$, or equivalently, $f_{ij} = f'_{ij} + \ell_{ij}$.

Then 1) may be written as

2)
$$\text{minimize} \quad \sum_{(i,\, j) \in A} d_{ij} \ell_{ij} + \sum_{(i,\, j) \in A} d_{ij} f'_{ij}$$

$$\text{subject to} \quad f'(i, N) - f'(N, i) = \ell(N, i) - \ell(i, N) \qquad \forall\, i \in N$$

$$0 \le f'_{ij} \le c_{ij} - \ell_{ij} \qquad \forall\, (i, j) \in A\ .$$

Note that $\sum d_{ij} \ell_{ij}$ and $\ell(N, i) - \ell(i, N)$ are constants. As many as $|A|$ constraints have been eliminated, and the problem may be solved as a minimum cost flow problem as follows:

Add a fictitious source s and sink t.

If $\ell(N, i) - \ell(i, N) > 0$, put an arc from s to i with $c_{si} = \ell(N, i) - \ell(i, N)$. Otherwise put an arc from i to t with $c_{it} = \ell(i, N) - \ell(N, i)$. The capacity of arc (i, j) in the new network is $c_{ij} - \ell_{ij}$, d_{ij} is unchanged. Let f be a minimum cost maximal s-t flow in the new network. To determine an optimal flow f^* for the original network, take $f^*_{ij} = \tilde{f}_{ij} + f_{ij} + \ell_{ij}$, where $\tilde{f}$ was the original flow.

That the transformation is valid is easily verified. We have

$$0 \le f'_{ij} \le c_{ij} - \ell_{ij} \iff \ell_{ij} \le f'_{ij} + \ell_{ij} \le c_{ij} .$$

Also
$$f'(i, N) - f'(N, i) = \ell(N, i) - \ell(i, N) \iff$$

$$f'(i, N) + \ell(i, N) - f'(N, i) - \ell(N, i) = 0 \iff$$

$$f(i, N) - f(N, i) = 0 .$$

Hence the feasible sets are just translations of one another, and since the objective functions are also translations, the problems are equivalent.

The above procedure may be used for sensitivity studies. Suppose an optimal solution is obtained, and some arc costs and capacities are changed, making the current solution infeasible. In this case, the minimum cost flow network determined in step 3 may contain negative cost arcs. The Bellman-Ford $O(n^3)$ procedure may be used to determine if negative cycles exist. If not, all arc costs may be made non-negative by the transformation $d_{ij} \to \pi_i + d_{ij} - \pi_j$, where π_i is the length of the shortest path from s to i (see [2]). If negative cycles do exist, they may be eliminated by repeated

applications of Bellman-Ford, utilizing as much information as possible from previous iterations. Another possibility is to apply a technique suggested by Ellis Johnson [7], which is described at the end of this paper. Once the negative cycles are eliminated, all costs may be made non-negative, and the problem solved using an efficient Path algorithm.

Example

We give a brief example of the transformation. The original network is shown in Figure 1a. Note arc (1, 2) has $f_{12} > c_{12}$. Figure 1b shows the new network with all flows zero. In Figure 1c the lower bounds have been eliminated. The problem has been transformed to a problem of sending 2 units from s to t at minimum cost. Often the required flow from s to t will be small, which will make the minimum cost flow problem easy to solve. Figure 1d gives the solution to that problem. To obtain the optimal flows, one adds the flows in Figure 1d and the lower bounds to the original flows.

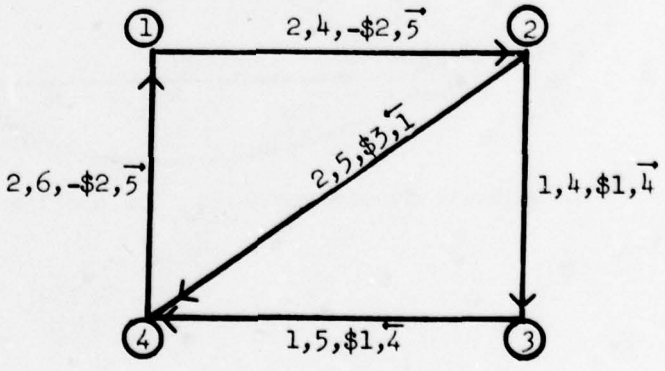


Figure 1a: The original network and starting infeasible flows. Numbers on each arc are ℓ_{ij}, c_{ij}, d_{ij}, f_{ij} in that order.

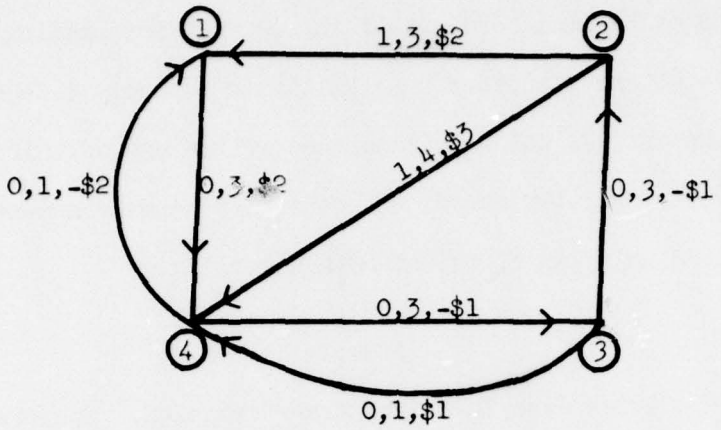


Figure 1b: The network for optimally augmenting the initial flow. All flows in this network are zero.

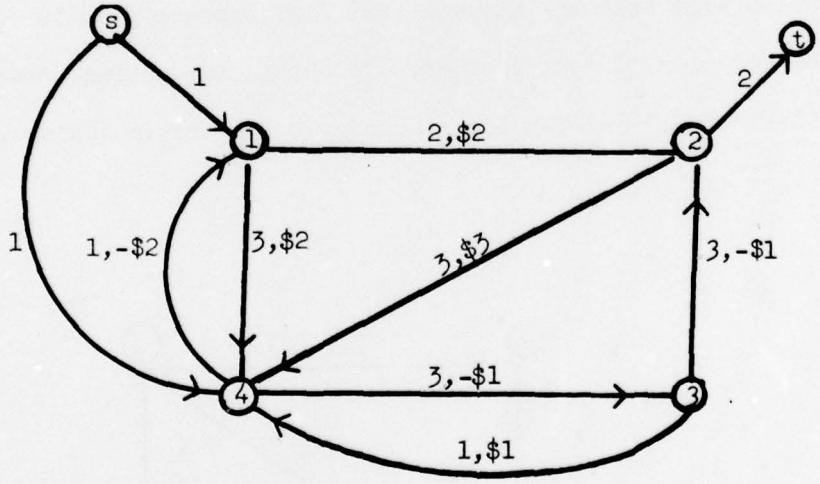


Figure 1c: The equivalent minimum cost flow problem.

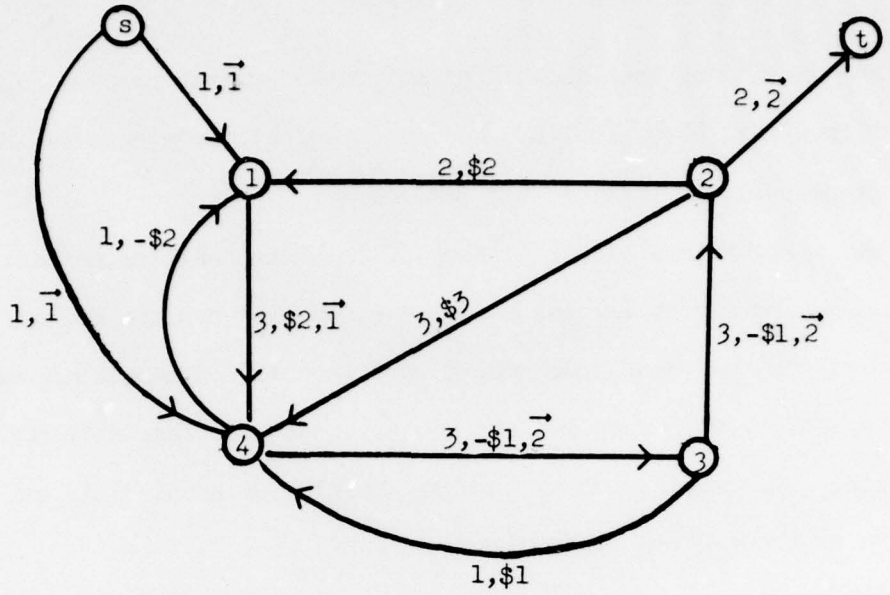


Figure 1d: Optimal solution to the minimal cost flow problem. Numbers with arrows represent flows. One unit of flow is sent over paths s432t and s1432t.

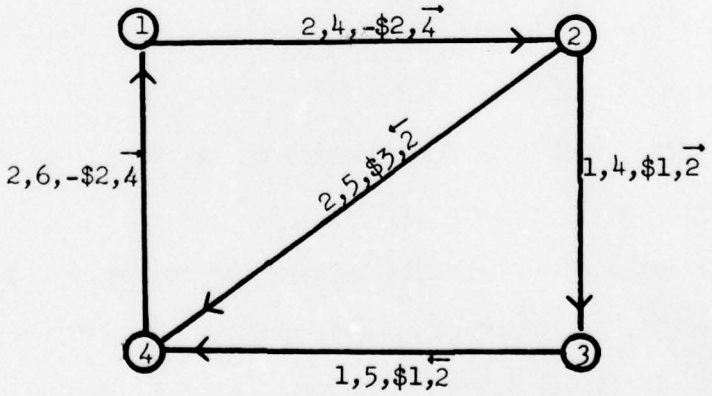


Figure 1e: Optimal flows for the original problem. Flows are obtained by adding lower bounds from Figure 1b and flows from Figure 1d to original flows.

Efficiency of the Out-of-Kilter Method

Thus far it has been shown that any Out-of-Kilter problem can be solved as a minimum cost flow problem. How do Out-of-Kilter's routines compare to other procedures for solving such problems?

Some insight into the efficiency of the Out-of-Kilter method may be obtained by looking at several examples which show how the method operates when solving a) max-flow problems, b) problems with non-optimal node numbers, or equivalently, with negative arc costs, and c) problems with non-negative arc costs. Parametric studies involve cheapest path calculations, and fall into one of these three categories.

Quick Review of the Out-of-Kilter Method

A feasible circulation will be optimal if it satisfies complementary slackness conditions which may be represented in the form of a kilter diagram as shown in Figure 2. A "node number" π_i may be thought of as the cost of the cheapest augmenting path from s to i, where (t, s) is the arc that is being brought into kilter.

Notice that with this interpretation, $\pi_i + d_{ij}$ is the cost of the cheapest path to j via arc (i, j).

The kilter diagram essentially says the following: In an optimal solution, if arc (i, j) is too expensive, meaning $\pi_i + d_{ij} > \pi_j$, we must have $f_{ij} = \ell_{ij}$. If (i, j) is cheap, meaning $\pi_i + d_{ij} < \pi_j$, we must have

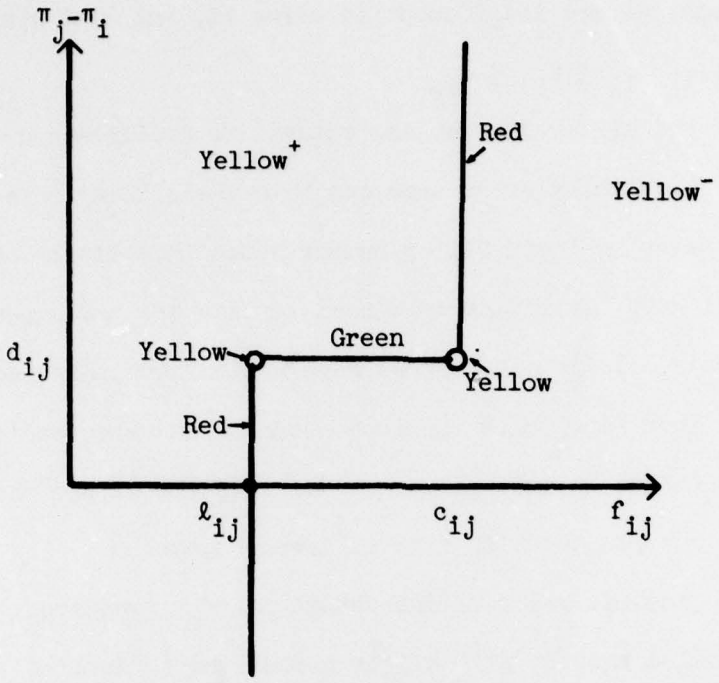


Figure 2: A graphical representation of the compementary slackness conditions. Points on the crooked line are in kilter. Regions are colored to facilitate discussion. The two yellow nodes have flows which <u>can</u> be increased in one direction but not the other.

$f_{ij} = c_{ij}$, meaning the cheapest path via (i, j) is blocked. If (i, j) lies on a cheapest path, we are indifferent to using it, and must simply have feasibility, i.e., $\ell_{ij} \leq f_{ij} \leq c_{ij}$.

Portions of the kilter diagram are colored to facilitate description of the algorithm. Essentially green arcs can have their flow increased or decreased without altering their kilter number since they lie on cheapest paths. Yellow$^-$ arcs must have their flow decreased as they are too expensive or have flow above capacity. Yellow$^+$ arcs must have their flow increased since they are too cheap or have flow below the lower bound. Red arcs can't have a flow change without a change in node numbers. Red arcs are either cheap with flow at upper bounds or expensive with flow at lower bounds.

The minimum absolute value of the change in f_{ij} required to bring (i, j) into kilter is called arc (i, j)'s kilter number and is denoted K_{ij}. The Out-of-Kilter method chooses an arc (t, s) which is not-in-kilter and reduces its kilter number to zero without increasing the kilter number of any other arc. This process is continued until all kilter numbers are zero. The procedures for reducing K_{ts} involve essentially either

a) computing a maximum flow from s to t (this computation is done in the network of all green and forward yellow arcs)

or b) computing a minimum cost augmenting path from s to t relative to arc costs which are modified by the $\{\pi_i\}$. (This computation is performed by setting the costs of all green and forward yellow arcs to zero (see Figure 3).)

How Out-of-Kilter Computes Shortest Paths

There are two types of yellow infeasible arcs which can be brought into kilter by increasing their flow:

1. Arcs with infeasible flow below their lower bound
2. Arcs with feasible flow that are cheap.

Figure 3 shows what happens when either type of arc is brought into kilter. To compute the shortest s-t path, arc (t, s) is added with $\ell_{ts} = 1$ to force 1 unit of flow from s to t. All flows, node numbers, and other lower bounds are zero.

To reduce K_{ts} (an example of case 1), Out-of-Kilter will treat all negative arcs as though they had zero cost, and then compute the shortest path from s to t using a $O(n^3)$ version of Dijkstra. This causes Out-of-Kilter to make an error by sending flow along the non-cheapest path s2t. Out-of-Kilter will then correct its error by augmenting around the negative cycle 1t2s1. This example can be made more pathological by increasing the capacity of s1t and providing numerous low capacity s-t paths of costs 3, 4, or 5 in a fashion similar to Figure 5.

If Out-of-Kilter initially chooses to reduce K_{1t}, it will introduce a reverse arc (t, 1) as shown in Figure 3c and then compute a shortest path from t to 1 in Figure 3c using $O(n^3)$ Dijkstra. Reverse arcs are not added in case 1 because the flow must be increased to have feasibility. In case 2 the node numbers across the arc may be modified to bring the arc into kilter without a flow change.

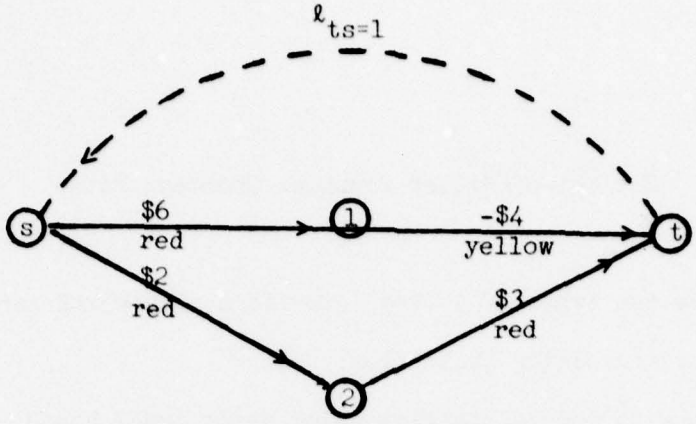


Figure 3a: Original network and arc colors.

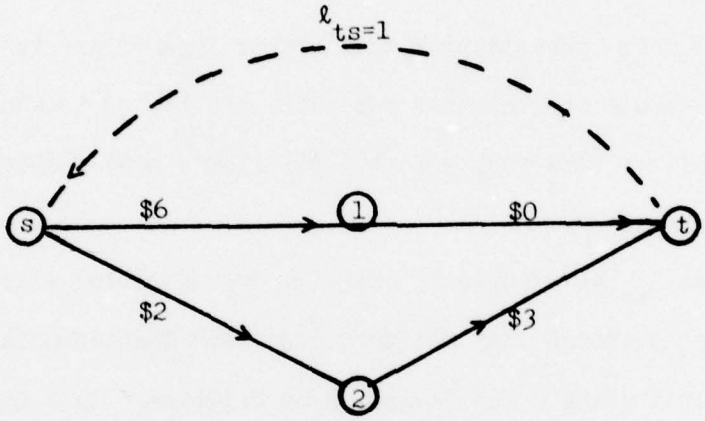


Figure 3b: Costs as viewed by Out-of-Kilter on first iteration if arc (t, s) is brought into kilter.

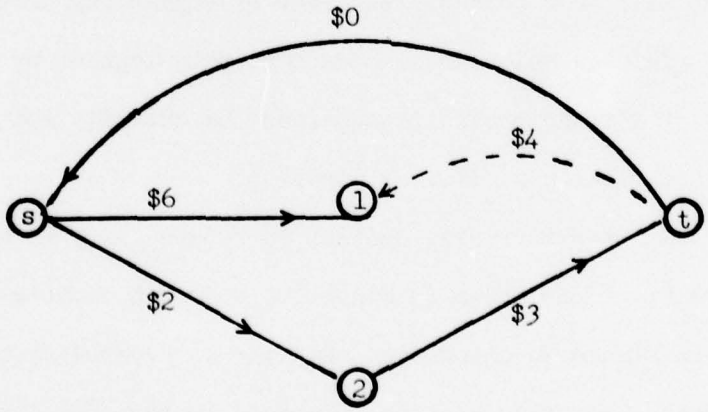


Figure 3c: Costs as viewed by Out-of-Kilter on first iteration if arc (1, t) is brought into kilter.

Figure 3: An explanation of how Out-of-Kilter does cheapest path computations. To reduce K_{ts}, Out-of-Kilter will compute a cheapest s-t path relative to costs shown in Figure 3b. To reduce K_{1t}, Out-of-Kilter will introduce the reverse arc (t, 1) and compute a cheapest path from t to 1 relative to costs shown in Figure 3c.

A Bad Maximum Flow Example

The following example is from [2]. When Out-of-Kilter is applied to the network in Figure 4, all arcs except (t, s) are in kilter. To bring (t, s) into kilter, Out-of-Kilter will perform a maximum s-t flow computation which could take 2×10^6 augmentations, since it uses the old Ford-Fulkerson labeling method which can alternate between paths s1267t and s5623t.

A Bad Example with Non-Optimal Node Numbers

The example in Figure 5 could be encountered during a sensitivity study. Node numbers $\pi_1 = 1$, $\pi_2 = 5, \ldots, \pi_{100} = 5$ are used to bias the arc costs so that Out-of-Kilter augments over non-cheapest paths. To reduce K_{ts}, Out-of-Kilter will paint (s, 1) red since its modified cost is $0 + 2 - 1 = 1 > 0$, and $f_{s1} = 0$. However arcs (s, 2),...,(s, 100) and (2, t),...,(100, t) will be yellow "can increase" since their modified costs are 0. Out-of-Kilter will send 1 unit of flow along paths s2t,...,s100t, and through arc (t, s), reducing its kilter number to 0. Arc (1, t) will still be out of kilter. To make K_{1t} zero, Out-of-Kilter will send 1 unit around cycles 1t2s1, 1t3s1,...,1t100s1. The optimal flow could have been found with one augmentation along the cheapest path s1t.

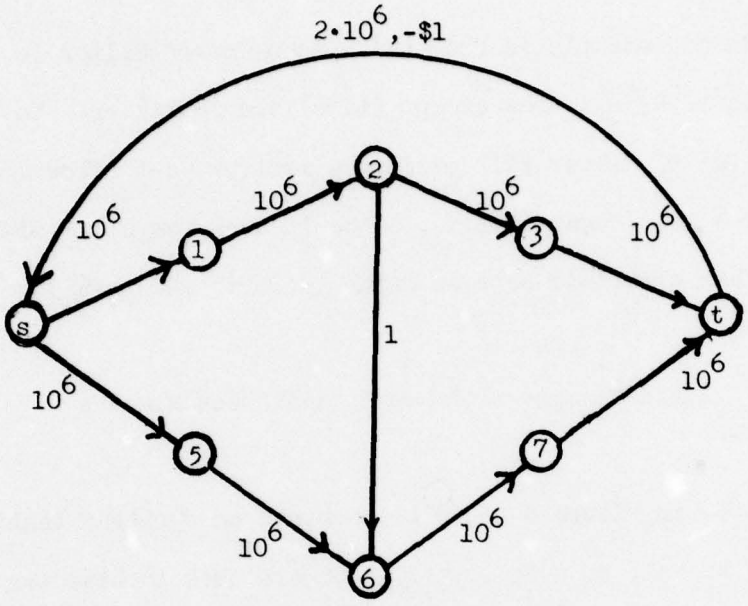


Figure 4: A pathological network for the Out-of-Kilter Method when applied to max-flow problems. Numbers shown are capacities. Arc (t, s) has a cost of -1. All other costs, flows, and node numbers are zero.

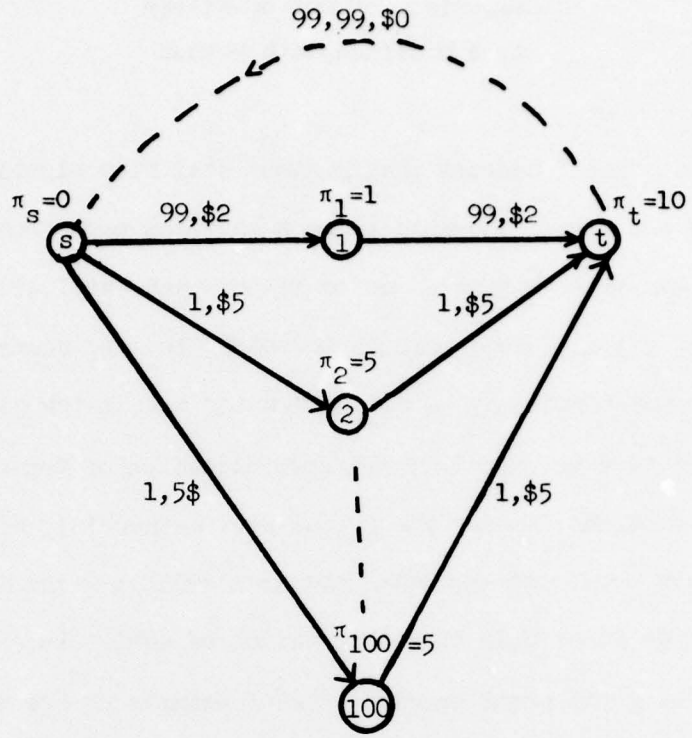


Figure 5: A pathological network for the Out-of-Kilter Method when applied to problems with non-optimal node numbers, or equivalently, negative arc costs. All flows and lower bounds on arcs other than (t, s) are zero. (t, s) has $\ell_{ts} = 99$ and $f_{ts} = 0$. Capacities and costs are as shown. To reduce K_{ts}, Out-of-Kilter will perform 99 augmentations over paths s2t, s3t,...,s100t. Then it will undo its work by performing 99 augmentations over the negative cycles slt2s, slt3s,...,slt100s, yielding the optimal solution, which could have been found with one augmentation along the cheapest path slt.

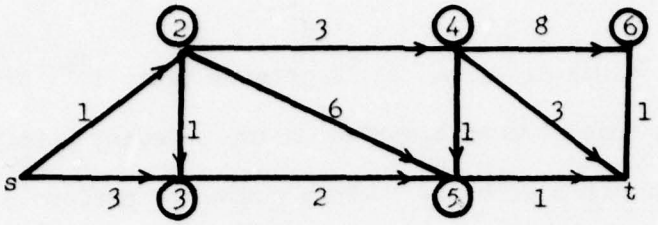
Comparison of Out-of-Kilter to a Modified Path Method

Recall that "Path" denotes that minimum cost flow algorithm which starts with zero flow and always augments along a cheapest path from source to sink. The algorithm we call "M-Path", due to Edmonds and Karp, also augments along cheapest paths, but uses node numbers to modify the arc costs to be non-negative so that the cheapest path may be computed using a Dijkstra calculation.

In this section we show that the specialization of Out-of-Kilter to minimum cost flow problems, namely the Primal Dual method [3], will essentially perform the same series of augmentations as M-Path, and produce similar node numbers. We first made this observation of equivalence in [12].

Before giving the proof we give a brief example of how the Dijkstra method and Primal Dual (Out-of-Kilter) compute node numbers (see Figure 6). In the first iteration, the shortest path from s to (2) is computed and both methods assign node (2) the permanent label 1. Dijkstra assigns node (3) the temporary label 3, Primal Dual gives all nodes on the right of the cut (s, 23456t) a temporary label of 1. In the next iteration, Dijkstra revises temporary labels of nodes adjacent to node 2 and labels node (3) permanently, while Primal-Dual adds 1 to the node numbers of all nodes on the right of the cut (s2, 3456t) and permanently labels node (3).

After the 4th iteration, the relevant portion of the network (for the maximum flow computation) consists of nodes s2345 and t. Node 6 is not



Initial Network (only costs are shown)

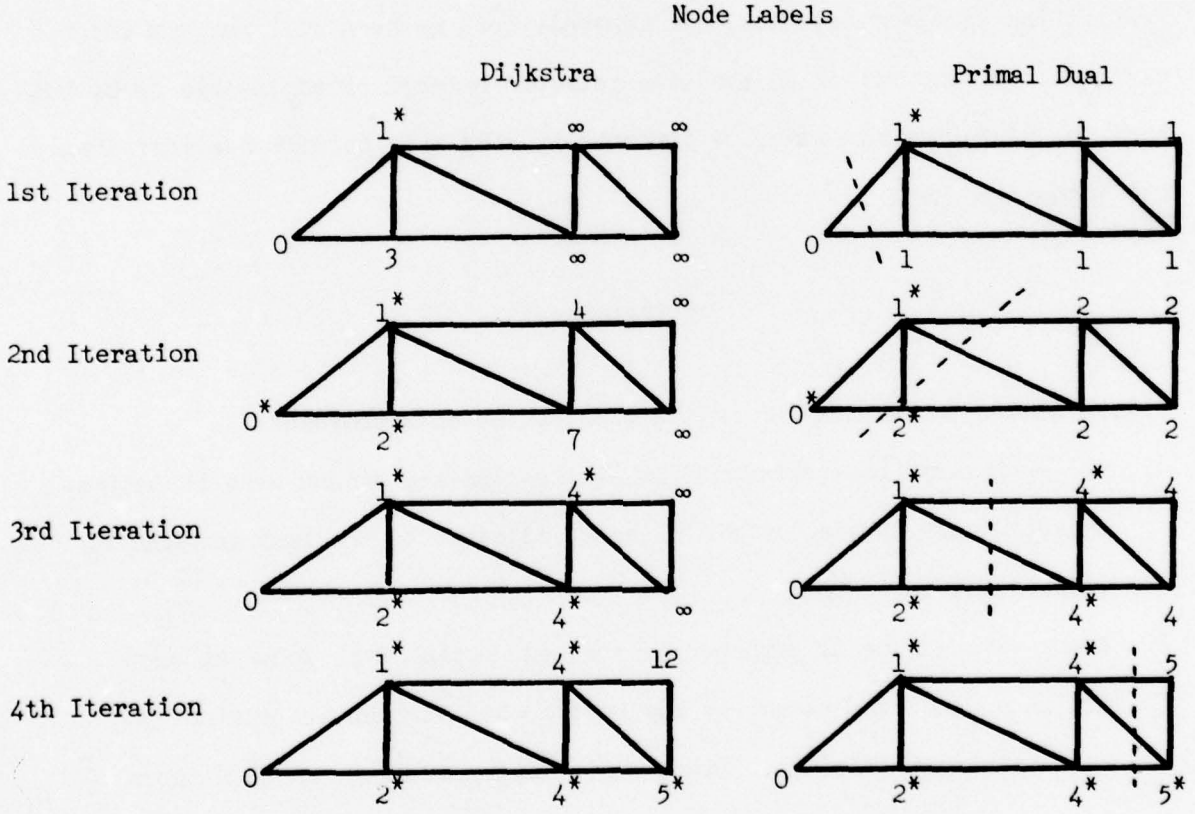


Figure 6: Comparison of Node Numbers Generated by Dijkstra and Primal Dual Methods

(Asterisk indicates permanently labeled node. Dotted line corresponds to cut in Primal Dual method.)

relevant since its "distance" from s is greater that t's distance from s. Notice that the node numbers computed on the relevant portion of the network are the same for both methods. M-Path would now perform a maximum s-t flow computation on the network of all shortest paths, resulting in one augmentation along the path s235t. Primal-Dual would compute a maximum s-t flow in the network of admissible arcs, the relevant portion of which is the same as that for M-Path, since it is easy to show that an arc is admissible if it lies on a shortest path from s to some node u. Thus both methods perform the same augmentation. In general, both methods will perform their maximum flow computation on the same relevant network of admissible arcs, but they may route the maximum flow differently when this network has more than one augmenting path.

Efficiency

Primal Dual's implementation of Dijkstra has several drawbacks.

1. All temporary (non-permanent) labels are updated during each iteration. Dijkstra only updates labels of nodes adjacent to the last permanently labeled node.
2. Every time a node is permanently labeled, Primal-Dual looks at each arc in a cut, even though it may have looked at that arc during the last permanent labeling. Since there may be as many as $n^2/4$ arcs in a cut, n permanent labelings yields a $O(n^3)$ algorithm. Dijkstra avoids this problem by not looking at the same arc twice.

Primal Dual's initial Dijkstra computation is $O(n^3)$. However, after the first augmentation, Primal Dual recomputes the set of nodes reachable by shortest paths fairly efficiently (using the Ford-Fulkerson labeling procedure), and thus can take between $O(n^2)$ and $O(n^3)$ steps per subsequent shortest path computation. To illustrate, the costs of a network are shown in Figure 7a and its admissible arcs in Figure 7b. Suppose an augmentation is made along the unique s-t path of admissible arcs, saturating both arcs of cost 2. At this point Primal Dual (Out-of-Kilter) would relabel all nodes reachable from s along paths of modified cost zero. This reconstruction of the network of shortest paths can be done quickly but involves some unnecessary recomputation which the Simplex method avoids by saving the unblocked portion of the shortest path tree. (More will be said about this in a forthcoming paper.)

Once Out-of-Kilter breaks across the cut using the dotted arc of modified cost 1 (see Figure 7d), it will try to label more nodes using the rest of the original admissible arc graph and thus may find a new cheapest path or paths after several breakthroughs. Properly implemented, this procedure can take less than n^2 steps per cheapest path calculation. However, as implemented by Out-of-Kilter, we show via Figure 8 that each successive cheapest path calculation can be $O(n^3)$.

Some of the problems discussed here have been eliminated in the improved version of Out-of-Kilter [1].

Numbers represent costs. Capacities are not shown.

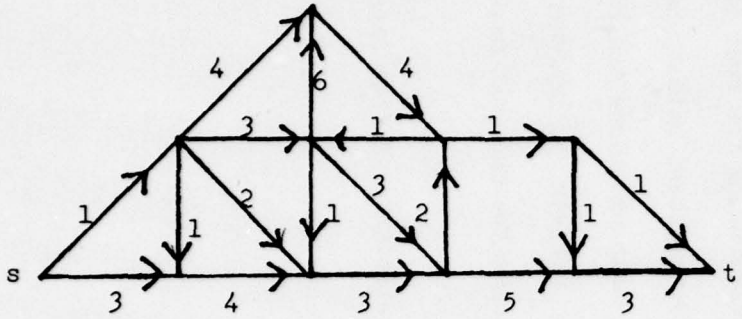


Figure 7a: The Original Network.

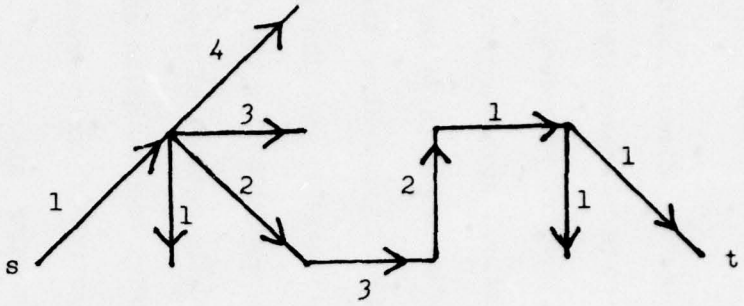


Figure 7b: Arcs on Shortest Paths.

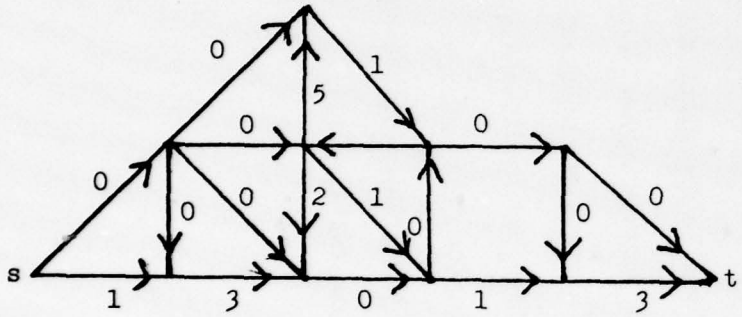


Figure 7c: Modified Costs.

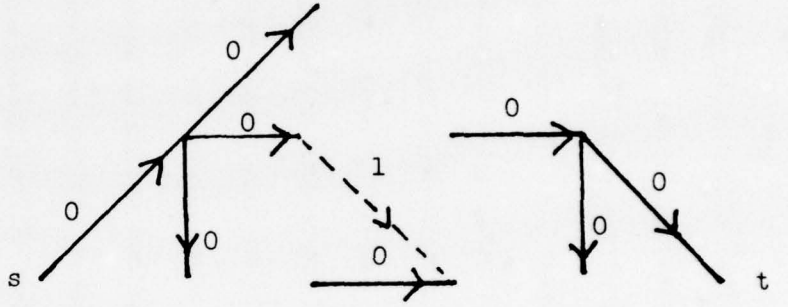


Figure 7d: Breaking across a Cut after an Augmentation which Saturates Two Arcs.

Figure 7: An Explanation of How Out-of-Kilter Iteratively Computes Shortest Paths.

Detailed Comparison of M-Path and Primal Dual

Recall that $d(u, v)$ denotes the cost per unit of flow in (u, v). Given a flow f in a network G, the network of possible ways of sending flow, called the augmentation network, is denoted by G^f. Let $\bar{d}(u, v)$ equal the minimum cost of an arc (u, v) in the augmentation network.

Both M-Path and Primal-Dual assume that all arc costs are non-negative, by making, if necessary, a cheapest path calculation in a network with negative costs, and modifying the $d(u, v)$ by the resultant node numbers. They also assume no negative cycles. This is reasonable, as augmentations around negative cycles can be performed until one or more arcs in each cycle are blocked.

Review of M-Path [2]

1) Set f^0, the initial flow, equal to zero.

2) Set π^0, the initial vector of node numbers, equal to zero.

3) Let f^k and π^k be the flows and vector of node numbers after the k^{th} maximum flow computation. To determine f^{k+1}, compute a maximum flow f which can be sent along minimal cost paths from s to t in G^{f^k} with respect to the non-negative costs $\Delta^k(u, v) = \pi^k(u) + \bar{d}(u, v) - \pi^k(v)$. ($\Delta^k(u, v)$ may be thought of as the minimum amount $\bar{d}(u, v)$ would have to be decreased to put (u, v) on a shortest path from s to v.) When computing f, always augment over a minimum cost path with the fewest arcs. Set $f^{k+1} = f^k + f$.

4) If $\sigma^k(u)$ denotes the cost of a cheapest path from s to u with respect to the costs Δ^k, set $\pi^{k+1}(u) = \pi^k(u) + \sigma^k(u)$. If u cannot be reached from s in G^{f^k}, set $\pi^k(u) = +\infty$.

5) Stop when the desired level of flow is reached.

It is shown in [2] that $\pi^k(u)$ equals the cost of a cheapest path from s to u in G^{f^k}.

Review of Primal Dual [3]

1) Set f^0, the initial flow, equal to zero, and set $\hat{\pi}^0$, the initial vector of node numbers, equal to zero.

2) Given $\hat{\pi}^k$ and f^k, compute $\hat{\pi}^{k+1}$ as follows:

   a) Take $S = \{s\}$, and take $\hat{\pi}^{k+1}(s) = 0$.

   b) At a general step, let S equal all nodes in G^{f^k} which are reachable from s by a path of admissible arcs, i.e., arcs (u, v) such that

$$(\hat{\pi}^k(u) + \Delta\hat{\pi}^k(u)) + \bar{d}(u, v) - (\hat{\pi}^k(v) + \Delta\hat{\pi}^k(v)) = 0 ,$$

   where $\hat{\pi}^k(u) + \Delta\hat{\pi}^k(u)$ is the node number of u at this stage. Determine

$$\delta = \min_{\substack{u \in S \\ v \in \sim S \\ (u, v) \in N^{f^k}}} \{(\hat{\pi}^k(u) + \Delta\hat{\pi}^k(u)) + \bar{d}(u, v) - (\hat{\pi}^k(v) + \Delta\hat{\pi}^k(v))\}$$

   and set $\Delta\hat{\pi}^k(v) = \Delta\hat{\pi}^k(v) + \delta$ for all $v \in \sim S$. Continue doing

b) until $t \in S$. At this point, set $\pi^{k+1}(u) = \pi^{k}(u) + \Delta\pi^{k}(u)$.

3) Determine f^{k+1} by adding to f^{k} a maximal s-t flow in $G^{f^{k}}$ $\cap$ all current admissible arcs.

Definition: The network of relevant admissible arcs in $G^{f^{k}}$ consists of those arcs which lie on a shortest path from s to some node u whose distance from s is not greater than t's distance from s.

Theorem 1:

The Primal Dual and M-Path methods are equivalent in the following sense: If both methods have computed the same sequence of flows $f^{1},\ldots,f^{k}$, then

1) Both methods will perform their $k+1$th maximum flow computation on the same relevant network of admissible arcs.

2) The permanent node numbers on this network will be the same for both methods, and will be computed in the same order.

Proof:

Let $\Pi^{k}=\{\text{nodes } u \mid \pi^{k}(u) \leq \pi^{k}(t)\}$. We will first show that both methods compute the same node numbers on Π^{1}, starting with $\pi^{0} = 0$.

It suffices to show that prior to the first maximum flow computation, the Primal Dual method will assign $u \in \Pi^{1}$ a node number equal to $\pi^{1}(u)$.

Primal Dual Computations

1) Initial computation

a) $S = \{s\}$

b) $\delta = \min_{u \neq s} \{d(s, u)\}$

c) All nodes $u \neq s$ are given node numbers equaling δ.

If $\delta = d(s, z)$, then arc (s, z) becomes admissible, and hence z is permanently labeled with the node number $d(s, z)$, because z will lie on the source side of all future cuts. Dijkstra also labels z permanently with the number $d(s, z)$.

2) In the general step, assuming inductively that all $u \in S$ have been labeled with $\pi^1(u)$, we obtain

$$\delta = \min_{\substack{u \in S \\ v \in \sim S \\ (u, v) \in G^{f^0}}} \{\pi^1(u) + d(u, v)\} - \pi ,$$

where π is the current label of all nodes in $\sim S$. If $\delta = \pi^1(w) + d(w, z) - \pi$, then z will be labeled with the node number $(\pi^1(w) + d(w, z) - \pi) + \pi = \pi^1(w) + d(w, z)$, exactly as in the Dijkstra method.

Thus it may be verified that prior to the first maximum flow computation, the node numbers generated by both methods for nodes in Π^1 are identical. This implies the set of admissible arcs are also identical.

Suppose inductively that Primal Dual and M-Path have computed the same node numbers $\pi^k(u)$ for $u \in \Pi^k$, and have computed the same flows f^k.

We must show that Primal Dual will compute the node number $\pi^{k+1}(u)$ for every $u \in \Pi^{k+1}$. Assume inductively that starting from $S = \{s\}$, the Primal Dual method has computed node numbers $\pi^{k+1}(u)$ for $u \in S$. At this stage, $\delta = \min \{\delta_1, \delta_2\}$, where

$$\delta_1 = \min_{\substack{u \in S \\ v \in \Pi^k \cap \sim S \\ (u, v) \cup G^{f^k}}} \{\pi^{k+1}(u) + \bar{d}(u, v) - \pi^k(v) - \pi\},$$

and

$$\delta_2 = \min_{\substack{u \in S \\ v \in \{\sim \Pi^k\} \cap \{\sim S\} \\ (u,v) \in G^{f^k}}} \{\pi^{k+1}(u) + \bar{d}(u, v) - \pi^k(t) - \pi\},$$

and where π equals the increase in the labels of all nodes in $\sim S$ since the computation of f^k. Suppose z is the next node to be permanently labeled by Primal Dual via node w. If $\delta = \delta_1$, then $\delta = \pi^{k+1}(w) + \bar{d}(w, z) - \pi^k(z) - \pi$, and z gets the permanent label $\delta + \pi^k(z) + \pi = \pi^{k+1}(w) + \bar{d}(w, z)$, just as in the Dijkstra method.

If $\delta = \delta_2$, we must show that the cost of any path P from s to z in G^{f^k} must be at least $\pi^{k+1}(w) + \bar{d}(w, z)$. Let (u, x) be the arc of P that lies in the cut $(S, \sim S)$. If $x \notin \Pi^k$, the result is clear. If $x \in \Pi^k$,

then since $z \notin \Pi^k$, cost (P) $\geq \pi^{k+1}(u) + \bar{d}(u, x) + \pi^k(t) - \pi^k(x)$

$$\geq \pi^{k+1}(u) + (\delta_1 + \pi^k(x) + \pi - \pi^{k+1}(u)) + \pi^k(t) - \pi^k(x)$$

$$\geq \delta + \pi + \pi^k(t) = \pi^{k+1}(w) + \bar{d}(w, z).$$

■

Sensitivity Analysis

The following example shows how one can make use of optimal node numbers when capacities are varied during a sensitivity analysis. Suppose one is given an optimal solution π^*, f^* to a minimum cost circulation problem, where π^*, f^* satisfy the complementary slackness, or equivalently, the kilter conditions. This is the same as saying that the modified costs $\pi_i + \bar{d}_{ij} - \pi_j$ of arcs (i, j) in G^{f^*} are all ≥ 0. Suppose now that the capacity of arc (u, v) with $f^*_{uv} = c_{uv}$ is decreased by 1. Then the problem becomes one of computing a cheapest path from u to v in G^{f^*}. Since all modified costs are ≥ 0, one can use Dijkstra immediately without any $O(n^3)$ shortest path computation.

If d_{uv} is increased by δ and c_{uv} is held constant, no improvement in the solution can be made as long as $\pi_u + d_{uv} - \pi_v + \delta \leq 0$, since complementary slackness will still be satisfied. When $\pi_u + d_{uv} - \pi_v + \delta > 0$, negative cycles containing (u,v) have been introduced and one can determine the new optimal solution by computing the shortest path from u to v in G^{f^*} using Dijkstra and applying a Path method until all flow on (u,v) has been rerouted or $\pi_u + d_{uv} - \pi_v^{new} + \delta = 0$, meaning (u,v) is now the current shortest path.

The case where $f^*_{uv} = \ell_{uv}$ and d_{uv} is decreased is similar.

When many arc costs are altered, there may be many negative cycles introduced. For this case, Ellis Johnson [7] has proposed a generalization of the above procedure which converts the problem of eliminating all the negative cycles into one minimum cost flow problem with non-negative arc costs. Johnson starts by setting the flow in all negative modified cost arcs up to capacity, and the flow in all positive modified cost arcs down to zero. Because the former solution was optimal, the arcs requiring such a flow change will be a subset of the set of arcs whose cost was changed. If f_{ij} is increased to c_{ij}, then node $j(i)$ is treated as a source (sink) with capacity (requirement) $c_{ij} - f_{ij}$. A reverse arc (j,i) is introduced with capacity c_{ij} and cost $-d_{ij}$, making its modified cost positive. Satisfying the source-sink requirements at minimum cost solves the problem.

Efficiency of Out-of-Kilter when Computing Shortest Paths in Graphs with Negative Arc Costs

The following example shows that Out-of-Kilter may take $O(n^5)$ steps to initially compute a shortest s-t path or find a negative cycle. Out-of-Kilter cannot take more than $O(n^5)$ steps since there are at most $O(n^2)$ negative arcs, and bringing each arc into kilter can take at most $O(n)$ breakthroughs, each of which may require looking at no more than $O(n^2)$ arcs in a cut.

The starting flows and node numbers in Figure 8 are zero. Arc (t,s) is added to put the problem in circulation form. In general, the two left most stacks may be taken to contain $\frac{n}{3}$ nodes each. The cost of arc (i,j) in a stack in general is $d_{ij} = -(\frac{n}{3}(i - 1) + 2(j - \frac{n}{6}) - 1)(\frac{n}{3} - 1)$ for $1 \leq i \leq \frac{n}{6}$, $\frac{n}{6} + 1 \leq j \leq \frac{n}{3}$, and $d_{ij} = d_{i'j'} - (\frac{n}{3} - 1)$ for $\frac{n}{3} + 1 \leq i \leq \frac{n}{2}$, $\frac{n}{2} + 1 \leq j \leq \frac{2n}{3}$, where $i' = i - \frac{n}{3}$, $j' = j - \frac{n}{3}$. The network is complete with all arcs not shown having extremely high cost.

Initially all arcs in the two left most stacks are Out-of-Kilter. Arc $(1,4)$ is brought into kilter, requiring a shortest path computation which will require $\frac{n}{3}$ breakthroughs and computational effort proportional to roughly $\sum_{i=1}^{[\frac{n}{3}]} i(n - i) \approx \frac{n^3}{20}$. In the present example the breakthrough process would stop at node 18 since $(1,4)$ would then be in kilter. Node numbers generated during this process are shown in Table 1. Arc $(7,10)$ is now brought into kilter, requiring in general a computational effort proportional to $\frac{n^3}{20}$, with the breakthrough process for the current example stopping at node 13. This process is continued until all $\frac{2n^2}{9}$ arcs in the stacks are brought into

kilter, for a total effort proportional to $\frac{n^5}{90}$. The essence of this example is that arcs are brought into kilter in such a way that Out-of-Kilter must keep rebuilding a large portion of the shortest path tree.

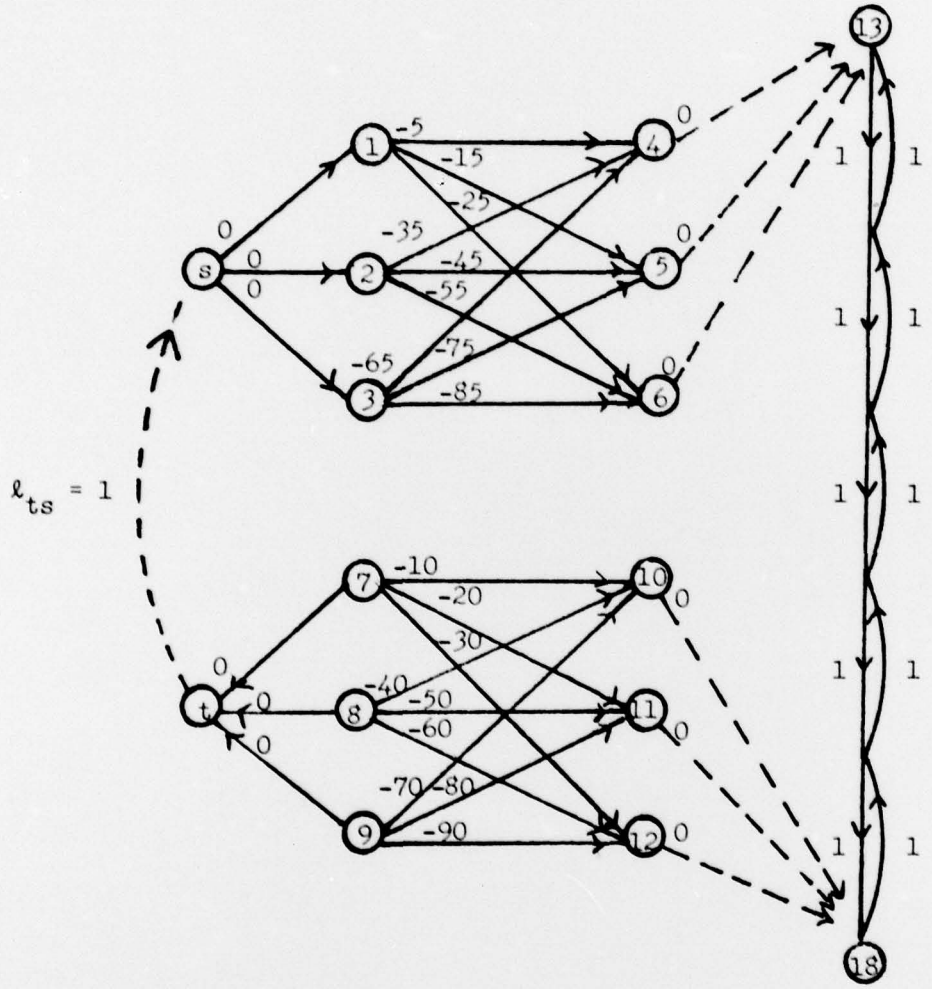


Figure 8: Construction of networks for which Out-of-Kilter may take $O(n^5)$ steps to compute a shortest s-t path.

The networks are complete. Arcs not shown have extremely high cost. In general there could be $n/3$ nodes in the top and bottom stacks and $n/3$ nodes in the path on the right.

Table 1

Node Numbers for the First Eight Arcs Brought into Kilter

Arc brought into kilter	(1,4)			(7,10)		
Node Numbers in Network	5	0	0		15	10	10
	5	5	5	1	15	15	15	9
		5	5	2		15	15	8
	5	5	5	3		15	5	7
		5	5	4	15	15	15	6
		5	5	5		15	15	5
Arc	(1,5)			(7,11)		
		30	25	15		50	45	35
	30	30	15	16	50	50	35	34
		30	30	17		50	50	33
		30	20	18		50	40	32
	30	30	30	19	50	50	30	31
		30	30	20		50	50	30
Arc	(1,6)			(7,12)		
		75	70	50		105	100	80
	75	75	60	51	105	105	90	79
		75	50	52		105	80	78
		75	65	53		105	95	77
	75	75	55	54	105	105	85	76
		75	75	55		105	75	75
Arc	(2,4)			(8,10)		
		135	100	100		165	130	130
	135	135	125	101	165	165	155	129
		135	115	102		165	145	128
		135	125	103		165	125	127
	135	135	115	104	165	165	145	126
		135	105	105		165	135	125

ACKNOWLEDGEMENT

The author would like to thank Professors Richard M. Karp and Eugene L. Lawler for their helpful comments.

REFERENCES

1. Barr, R. S., F. Glover, and D. Klingman, "An Improved Version of the Out-of-Kilter Method and a Comparative Study of Computer Codes," Mathematical Programming 7(1974) 60-86.

2. Edmonds, J., and R. M. Karp, "Theoretical Improvement in Algorithmic Efficiency for Network Flow Problems," J. A. C. M. 19 (1972), 248-264.

3. Ford, L. R., Jr., and D. R. Fulkerson, "A Primal Dual Algorithm for the Capacitated Hitchcock Problem," Naval Res. Logist. Quart. 4 (1957), 47-54.

4. Ford, L. R., Jr., and D. R. Fulkerson, Flows in Networks, Princeton University Press, Princeton, New Jersey, 1962.

5. Fulkerson, D. R., "An Out-of-Kilter Method for Minimal Cost Flow Problems," SIAM J. Appl. Math., 9 (1961), 18-27.

6. Glover, F., D. Karney, and D. Klingman, "Implementation and Computational Comparisons of Primal, Dual, and Primal-Dual Computer Codes for Minimum Cost Network Flow Problems," Networks 20, 191-212 (1974).

7. Johnson, E. L., "Flows in Networks," in Handbook of Operations Research, J.J. Moder and S. E. Elmaghraby editors, Van Nostrand Reinhold & Co. (1978), pp. 183-206.

8. Lawler, E., Combinatorial Optimization, Holt, Rinehart, and Winston, (1976), pp. 206-207.

9. Minty, G. J., "Monotone Networks," Proc. Roy. Soc. London, Ser. A, 257 (1960) 194-212.

10. Murty, Katta, "Linear and Combinatorial Programming," John Wiley and Sons, (1976), pp. 360-381.

11. Shapiro, J. F., "A Note on the Primal-Dual and Out-of-Kilter Algorithms for Network Optimization Problems," Networks 7, (1977), 81-88.

12. Zadeh, N., "Theoretical Efficiency and Partial Equivalence of Minimum Cost Flow Algorithms: A Bad Network Problem for the Simplex Method", ORC Report 72-7, University of California at Berkeley (1972).

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 35	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Simple Alternative to the Out-of-Kilter Algorithm		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Norman Zadeh		8. CONTRACT OR GRANT NUMBER(s) N00014-75-C-0493
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Operations Research Stanford University Stanford, California 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS (NR-042-264)
11. CONTROLLING OFFICE NAME AND ADDRESS Logistics and Mathematical Statistics Branch Office of Naval Research Arlington, Virginia 22217		12. REPORT DATE May 31, 1979 13. NUMBER OF PAGES 35
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale.
Its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Also partially supported by National Science Foundation Grant ENG-76-12266.

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Out-of-Kilter algorithm, network flows, shortest paths, transportation problems.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

See reverse side.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

A Simple Alternative to the Out-of-Kilter Method

Abstract

It is shown that any problem solvable by the Out-of-Kilter method may be simplified to an ordinary minimum cost flow problem (meaning a problem with zero lower bounds). To perform the simplification, one considers the equivalent problem of optimally augmenting the original flows and then eliminates the lower bounds from this problem. In linear programming terms, as many as $|A|$ constraints are eliminated without adding new variables, where $|A|$ is the number of arcs.

When the simplified problem has non-negative arc costs, we show that Out-of-Kilter will solve it by augmenting along a sequence of shortest paths, and hence is just another implementation of such methods. In particular, Out-of-Kilter combines the Ford-Fulkerson labeling procedure with a $O(n^3)$ version of Dijkstra that computes shortest paths by setting the costs of negative arcs to zero.

Examples are presented which show that Out-of-Kilter may behave pathologically when some arc costs are negative. For example, Out-of-Kilter may take $O(n^5)$ steps to find a negative cycle or compute a shortest path.

The following procedure is suggested to replace the Out-of-Kilter method: Transform to a minimum cost flow problem, eliminate negative cycles if any, then efficiently augment along a sequence of shortest paths.